A Novel P2P and Cloud Computing Hybrid Architecture for Multimedia Streaming with QoS Cost Functions

Irena Trajkovska
Technical University of Madrid
DIT, ETSIT, UPM, Spain
irenatr@dit.upm.es

Joaquín Salvachúa Rodrígues
Technical University of Madrid
DIT, ETSIT, UPM, Spain
jsalvachua@dit.upm.es

Alberto Mozo Velasco
Technical University of Madrid
EUI, UPM, Spain
amozo@eui.upm.es

ABSTRACT

Since its appearance, peer-to-peer technology has given raise to various multimedia streaming applications. Today, cloud computing offers different service models as a base for successful end user applications. In this paper we propose joining peer-to-peer and cloud computing into new architectural realization of a distributed cloud computing network for multimedia streaming, in a both centralized and peer-to-peer distributed manner. This architecture merges private and public clouds and it is intended for a commercial use, but in the same time scalable to offer the possibility of non-profitable use. In order to take advantage of the cloud paradigm and make multimedia streaming more efficient, we introduce APIs in the cloud, containing build-in functions for automatic QoS calculation, which permits negotiating QoS parameters such as bandwidth, jitter and latency, among a cloud service provider and its potential clients.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Applications

General Terms

Design

Keywords

P2P, QoS, cloud computing, hybrid architecture, streaming

1. INTRODUCTION

Soon after P2P file sharing market reached the fame, a new way of distributing the content was introduced - streaming (real-time and on-demand) on the top of the P2P overlay that attracted many users to switch to a new way of watching the media content with no need to download.

Today, Cloud Computing (CC) has placed a serious strain on the business market. In the same time, it inspired researches to think of integrating an already existing technologies with cloud computing, taking the advantages CC offers.

M. Fouquet et al.[8], present a good overview of the utilities CC offers. They describe a distributed application-layer multicast for video streaming engined by relay servers, discussing its possible integration with the cloud. They present in details the distribution trees topology, numbering various use cases of how to benefit from the cloud. Latency and bandwidth issue in video streaming is mentioned without further details.

Therefore as principal objective, we consider building automatic API functions based on Quality of Service (QoS) parameters to leverage a novel hybrid cloud and P2P architecture for multimedia streaming. We favour cloud paradigm to be future key for bringing back the Client-Server (CS) topology in multimedia communication, and by expanding it for P2P streaming support we believe it could bring double benefit to both the cloud service providers, and the end users.

The presented architecture offers a transparent approach towards QoS guided business model. Organized as a multimedia streaming distributed overlay, it presents an idea for application interface intended for providers of multimedia streaming content. The Isabel [4] platform for videoconferencing service includes a feature in its infrastructure, that registers and stores streaming packet time stamps of the participants in a video conference.

We suggest that by extending this feature it is possible to implement automatic functions for calculation of QoS parameters (bandwidth, jitter and latency). The functions would be part of a web service and will be represented to the cloud providers' clients through a user friendly interface. This would enable them by connecting to the provider's page, consult the current status of a streaming content, the connected clients using the service and their QoS status.

The API would furthermore offer a price model, that permits straightforward decision on whether to watch streaming in a CS or P2P manner. Unlike the models for ISP guided choice of peer selection, we suggest a new and completely independent model for selection guided by clients' preferences for QoS parameters and price packets. This encourages the clients to take maximum advantage of the model guaranteed with QoS parameters, while the provider has complete monitoring of the dynamic QoS changes, therefore could react on time to reinforce its resources for better scalability.

This paper is organized as follows; In section 2 we discuss related work followed by an overview of the suggested architecture and description of the API functions in section

3. Section 4 focuses on calculation of QoS parameters, and section 5 summarizes and traces the way for future research.

2. RELATED WORK

No doubt exists for the achievement P2P technology has brought, as many applications, topologies and protocols have marked its maturity over the years. Implemented as an organized overlay in order to overcome the CS based limitations for expensive bandwidth provision cost, P2P attracted various open source and commercial designs for content sharing, Video on Demand (VoD) and live streaming. These have proven P2P's model increased contribution for scalability and robustness. BitTorrent as a pioneer in this area, is still leader among the P2P file-sharing applications.

Cloud computing brings back in game the virtual centralized model. The scalability and robustness of the cloud infrastructure offered by leading cloud providers as Amazon, Google, IBM, Oracle, leverage various commercial applications to move in the cloud. Multimedia live streaming and VoD Software as a Service, (SaaS) [6, 5] have already started to give their bet over the cloud infrastructure, in this case Amazon EC2 [2] and Cloud Front [1].

Cloud computing profit oriented nature could render more difficult the establishment of a mutual negotiation policy when combined with other non-profitable platforms. An example that deals with the long lasting battle between the P2P overlay networks and the ISPs is described in [7, 9, 10].

Based on locality-awareness method, overlay topology construction approach or on both, in order to reduce the cross-ISP traffic they suggest various methods such as oracle web service offered by the ISP, adaptive peer selection algorithm, algorithm based on a gossip protocol or hybrid approach adapting between biased and random peer selection. These techniques aim to facilitate a P2P node to choose neighbours independently based on a certain criteria. Last example presents an algorithm for peer-assisted VoD in which peers bias transmission rates to their neighbours, and dynamically adjust these rates.

The research work mentioned so far treats the trade off between the ISP and the P2P overlays build on the top of their network. However comparing among commercial cloud applications, no ISP's uniform payment policy for multimedia streaming and VoD service exist. Offers vary depending on the required cloud disc space or CPU for PaaS and IaaS solutions, while SaaS providers define their own cost models depending on the services they offer.

3. ARCHITECTURE OVERVIEW

Cloud commercial multimedia streaming applications are expected to meet certain QoS parameters. Yet no concrete pricing model based on automatic QoS parameters exists for commercial use. We wanted to explore this possibility and define price packets for customers of multimedia streaming service by considering three type of QoS parameters such as jitter, latency and bandwidth.

We propose a hybrid architecture for multimedia streaming that offers API functions based on QoS parameters. Furthermore, we put on scene the price politics for the commercial CS part with an alternative non-profitable P2P approach.

The architecture takes advantage of the CC scalable infrastructure that facilitates deployment of stable and robust services. An idea is offered to the users of the cloud infrastructure such as Autonomous Servers (AS) or ISPs, for a SaaS deployed as a web service application on the cloud.

Figure 1 represents a general view of the proposed architecture. It depicts the cloud that contains multimedia streaming servers, (their number depends on the provider of the streaming service with possibility to scale as the number of customers or petitions rise). The service has *first level* or directly connected clients (Client C1, C2 , C3,..C6) and *higher level clients* (Client C4_1, C4_2, C5_1 and C5_2). The idea behind this is that first level clients after login, consult and choose one among the three types of price packets. Afterwards, they decide to make contract directly with the provider of the service enjoying high streaming quality.

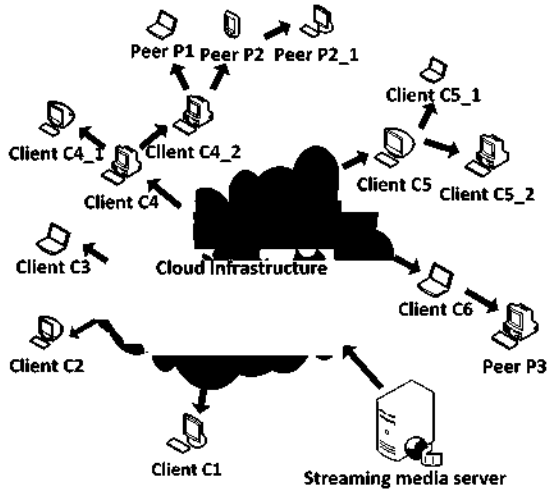

Figure 1: CS-P2P Streaming Cloud Architecture

Similarly, higher level are clients that by consulting API functions obtain an information list with QoS status for all connected clients and have decided to which client to connect. By doing this they agree on lower streaming quality (determined in the price model for higher level customers) and contract the first (or higher level) customers instead of the provider. There exists possibility for peers to connect to higher level clients who want to offer their service for free (Peer P1, P2, P2_1, P3).

This way the streaming topology is organized in a tree-based manner but it does not exclude the possibility to adapt it for a mesh-based or other convenient overlay. However, discussing types of overlay topologies are out of the scope in this paper.

The service provider has direct centralized management of the contract politics among all types of customers. It also takes control over the streaming and stored content on the server with possibility to contract external server for single or short term streaming. This may appear as a result of increased petitions for live or on demand streaming content.

Such a cooperation with the service provider acting as a mediator among third party servers and own clients, contributes the business as more streaming contents become available. The owner of an external streaming server profits from the cooperation with the cloud service provider and indirectly disposes with the same user friendly interface which attracts a number of new clients. Also it uses a stable service that relies on a scalable cloud computing platform of-

Table 1: Price Packets and Users' QoS

QoS	UserID: BV(Mbps), LV(ms), JV		
Gold	C1: 25, 20, 0.0435	C8: 19, 10, 0.5366	C4:..
Silver	C5: 13, 40, 1.4785	C9: 11, 60, 1.3299	C6..
Bronze	C2: 4, 80, 2.3401	C5: 5, 90, 2.7832	C7:..

fering higher bandwidth, lower latency, better load balancing, scalability and robustness. Aiming to figure out possible cost models between various external servers and SaaS providers on the cloud could be possible research work that could emerge from this idea.

Following the idea of Amazon Reserved Instances [3] and payment methods some telecommunication companies offer for various services, we introduce a cost model offered by the cloud provider of the streaming service to the end users. We foresee that it could possibly serve as a base to build uniform model for coordination among various service providers of multimedia streaming, on the one hand and cloud provider companies on the other hand. Such a user friendly price model will enable deployment of various SaaS in the cloud that can profit out of the dynamic QoS parameters displayed for all subscribed clients, and would permit to negotiate price among different clients.

We'll continue describing the API functions following the event flow scenario.

Users login to the service provider page (web service), willing to watch some streaming program or content on demand.

After choosing a content, the user invokes web service API function (1) that returns a table with all currently subscribed clients watching the specified `streaming content`. Call of function (2) would return a table, with type of price packets and users of each packet watching the `streaming content`.

```
(1) get_user_table(streaming content)
(2) get_price_packets(streaming content)
```

Shorten version for this output is shown on Table 1. Next to the user ID appear the three values of QoS parameters for video streaming packets. Price packets are defined regarding the QoS bounds guaranteed to the user of the streaming service. We suggest a generic model containing three types of price packets: Gold, Silver and Bronze.

Gold packet is most expensive offering best quality streaming with the highest bandwidth and least possible jitter and latency. **Silver** and **Bronze** offer lower quality specifically adapted for users who do not insist on the highest quality, but could rather tolerate lower QoS at a lower cost. The service could be set up on demand in minutes, hours, per streaming content etc., which together with the price of the packets is determined by the provider.

Besides direct CS connection, clients are allowed to connect to other clients and extend the streaming tree. New users can obtain information for connected clients invoking the associated API functions. Calling (3), would return all the clients of a concrete streaming content and their current QoS values. One can require a view of all peers registered in the session by calling (4).

```
(3) get_clients(streaming content)
(4) get_peers(streaming content)
(5) get_bandwidth(streaming content)
```

API (5) gives a list of all clients and their current values for bandwidth. Same API would appear for latency and jitter respectively. It could be also possible to combine API functions or specify desired values for some parameters.

```
(6) get_bandwidth_and_latency(streaming content)
(7) get_peers(bandwidth, jitter, latency)
```

Table 2: User Status Table

QoS	C	C-C	C-P	C-C,P	P	P-P
User ID	C1	C9	C5	C2	C10	C12
B(Mbps)	25	11	5	4	2	1
L(ms)	20	60	90	80	90	110
J	0.04	1.32	2.78	2.34	3.03	3.44

```
(8) get_direct_clients_of_packet(bandwidth, jitter,
    latency, packet)
(9) get_clients_status(streaming content)
```

Function (6) returns a list of clients together with their bandwidth and latency values, while (7) lists all peers who share the specified values for `bandwidth`, `jitter` and `latency` (some of those could be left empty if not required). (8) is complex function returning list of direct clients with specified QoS who have contracted `packet` with value gold, silver or bronze.

Having this information a new user can choose, either to connect to the provider and become its first level client, or connect to other client in the streaming tree depending on their QoS and price preferences. If none of those it could simply connect as peer and start streaming for free. Once the choice is made it could notify the provider, who takes care of the payment method.

With this, potential clients and peers have transparent view of the status of some streaming content with respect to the QoS type, their value intervals, the prices assigned to each interval including the clients and their status. This allows them, to choose depending on the QoS, the desired type of contract or connection. Another API function (9) would return the Table 2 containing the status of clients sorted according to the service they offer.

The first row contains letters to denote possible status combinations. The first letter in a tuple denotes if user identified with its ID number is client or peer, and the second one denotes whether they permit their service to other clients, peers or both. In a case they do not offer further streaming, the second letter is dropped.

For ex. $C9(c-c)$ denotes user $C9$ is client and permits only clients connected, or $C2(c-c,p)$-user $C2$ is client and permits both clients and peers. Note the absence of category $ID(p-c)$ for obvious reasons that once an user becomes a peer, it is not permitted to offer service to clients, due to low QoS values. Anyway this is regulated as no price is established for very low QoS and hence, a client(peer) with such values for QoS parameters could offer service only to peers further in the P2P tree.

On the other side it could be feasible by the third, fourth etc., level clients, i.e., sub-users of silver or bronze packets, to offer their service for further P2P redistribution of the streaming content due to low QoS values. For example, lets say that the client C of bronze packet has the QoS values for video (B: 2Mbps, L: 90ms, J: 2.55); this user has no befit to charge another client(s) for its service therefore sets up his status to $C(c-p)$ extending the tree with P2P streaming only.

A possible incentive scenario for the $c-p$ status could be a special offer by the provider as a price packet or discount, for motivating clients to offer streaming to peers that way stimulating P2P expansion and also contributing the CS model such that instead of being high level clients, they could connect directly to the provider at a lower rate then the rest of the first level clients.

Each client/peer decides their own status. As noted earlier the goal is to union CS and P2P streaming under the same model, so that everybody can benefit from the architecture. Once a choice is made the provider assigns the client an ID number, they pay for the service, set up the status and start streaming.

Figure 2 depicts reduced streaming scenario including a table as a part of the interface that contains information classified according to the bandwidth intervals. Each column contains QoS value interval as upper and lower bounds of the bandwidth and the price assigned to that interval. Below each price is a view of all currently connected clients under that category together with their status. If new user finds an interval and its assigned price convenient, it chooses to which client(peer) of that category to connect, depending on their current QoS values.

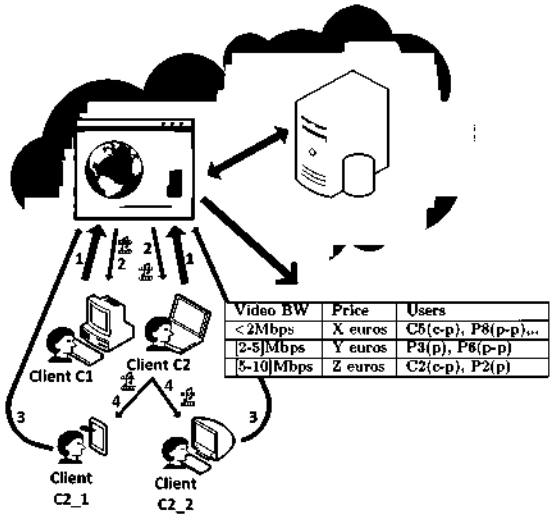

Figure 2: Example Topology with QoS Table

The cloud architecture as described enables better centralized control of the involved clients and peers. For example, in order to avoid service abuse, frauds by peers and prevent clients to offer service setting personal random prices, all contracts are established through the cloud control system. Payment between new and a current client will be purchased through a web service payment method after login. At the same time the QoS interface reflects current state of the resources which permits the service provider online control and monitoring with possibility to turn on or contract new cloud servers for scalability requirement. This would reduce latency and contribute higher robustness.

As for the churn issue, once a client pays the service it is in their best interest to stay connected until the end of the streaming due to the payment acquisition. However, legal commitments should be specified in the SLA for higher level clients who want temporally or permanently to leave the session. The service should then redirect the affected clients to other clients with status type $ID(c-c)$ from the same price rang. Churn among peers does not directly affect the global integrity of the architecture.

Currently it is very difficult to combine client-server and peer-to-peer streaming within the same service, mostly because of cost policies that require accurately established model. We believe to be the first who combine these two technologies under the umbrella of the cloud because we recognize high potential in the cloud infrastructure, especially regarding precise cost models that would facilitate future reinforced interaction between end users and cloud service providers.

4. CALCULUS OF QOS PARAMETERS

The QoS values mentioned in the previous section, are based upon an automatic calculation algorithm to be deployed. We base the packet time stamps acquisition on a feature in the Isabel platform for videoconferencing service and extend it to deploy the described APIs. The idea is described on Figure 2.

As a streaming session initiates, time stamps of outgoing and incoming audio and video packets are registered. The received data is stored on the provider database server and on the clients' machines. We will consider only time stamps of outgoing packets assuming the processing packet time very small, thus irrelevant for the calculation. Following are the well known formulas for QoS calculation as they would be used in our scenario, provided we have included the time stamp feature in the web service logic:

We observe Client C2_2 on Figure 2. If t_{C22} is packet time stamp at Client $C2\_2$, then its latency will be calculated by summing all the previous latencies from the server up to client $C2$ in the tree, adding the new time stamp for Client $C2\_2$, or $latency(C2\_2) = t_{C22} + latency(C2)$, where $latency(C2)$ is the latency of client $C2$ for the same packet. Latency for both audio and video packets is calculated according to the same formula and so for every new user in the streaming tree.

The jitter represents the statistical variance of RTP data packet inter-arrival time. Therefore if Si is time stamp for packet i, and Ri is the time of arrival in RTP time stamp units for packet i, then for two packets i and j we have the difference $D(i,j) = (Rj - Ri) - (Sj - Si) = (Rj - Sj) - (Ri - Si)$. The inter-arrival jitter of any client is calculated instantaneously as the data packets are received by the client using the difference D for that packet i and the previous packet $i-1$ according to the formula $J(i) = J(i-1) + \frac{(|D(i-1,i)| - J(i-1))}{16}$. The division by 16 is in order to reduce noise.

Finally to calculate the bandwidth we assume that the streaming packet size is constant during the streaming session. Supposing that the streaming channel is asymmetric we will consider only the upload bandwidth as of interest for new users. If the streaming bit rate br is known, depending on the audio/video codec used, the bandwidth of Client $C2$ is simply calculated as $bandwidth(C2) = n * br$, where n is the number of clients to which peer $C2$ is forwarding the streaming packets (two in this case). The bandwidth can be also calculated as $bandwidth(C2) = (ps_a * n * p_a/t_a) + (ps_v * n * p_v/t_v)$, where ps_a and ps_v are packet size of the audio and video the stream respectively (ex. bits/packet). Analogically p_a/t_a and p_v/t_v are the audio and video packet rates (ex. packet/time). The web service registers every time stamp value and assigns it to the equations. This way, is achieved a dynamic and more accurate QoS representation.

5. OPEN ISSUES AND FUTURE WORK

In this paper we presented a novel use case of the cloud infrastructure, introducing an architecture for P2P multimedia streaming in both CS and P2P style. At the same time we offered QoS API functions implemented in a web service of the cloud streaming service provider. This would enable users to decide about the contract type to establish with the service provider or go for P2P streaming as a possible solution. We believe this idea fills the gap between the long battle among P2P and ISPs, offering a transparent solution for both the service providers and the end users. Moreover it opens the door for the providers of streaming services to take the advantage of the cloud paradigm in order to deploy scalable SaaS with guarantees for QoS.

We leave as an open issue, implementation of the described APIs in a cloud infrastructure for near future by extending the Isabel system functionality and test its behaviour. As an interesting use case to explore, could be the extension of this architecture to support videoconferencing in the cloud. Other work could focus on the possibility to unify cost policies for cloud streaming services build on different cloud infrastructures.

6. REFERENCES

[1] Amazon CloudFront. `http://aws.amazon.com/cloudfront`.
[2] Amazon EC2. `http://aws.amazon.com/ec2`.
[3] Amazon Reserved Instances. `http://aws.amazon.com/ec2/reserved-instances`.
[4] Isabel. `http://isabel.dit.upm.es/content/view/16/34`.
[5] Musana. `http://musana.com`.
[6] Wowza. `http://www.wowzamedia.com/ec2.html`.
[7] V. Aggarwal, A. Feldmann, and C. Scheideler. Can ISPs and P2P users cooperate for improved performance? *SIGCOMM Comput. Commun. Rev.*, 37(3):29–40, 2007.
[8] M. Fouquet, H. Niedermayer, and G. Carle. Cloud computing for the masses. In *U-NET '09: Proc. of the 1st ACM workshop on User-provided networking: challenges and opportunities*, pg. 31–36. ACM, 2009.
[9] Z. Shen and R. Zimmermann. ISP-friendly peer selection in P2P networks. In *MM '09: Proc. of the 17th ACM Int. Conf. on Multimedia*, pg. 869–872. 2009.
[10] J. Wang, C. Huang, and J. Li. On ISP-friendly rate allocation for peer-assisted VoD. In *MM '08: Proc. of the 16th ACM Int. Conf. on Multimedia*, pg. 279–288. 2008.